IN THE CLAIMS:

Please amend the claims as follows:

1. (Currently Amended) A graphics processor for multithreaded execution of program instructions associated with threads to process at least two sample types comprising:

at least one multithreaded processing unit that includes

a thread control unit including:

a thread storage resource configured to store thread state data for each of the threads to process the at least two sample types, wherein the at least two sample types are assigned to the threads for multithreaded processing based on an allocation priority among the at least two sample types, and wherein a number of locations in the thread storage resource allocated for each of the sample types is determined and limited by using a sample portion global state value <u>which determines a number of each of the at least two sample types available for processing by the threads of the multithreaded processing unit</u>.

2. (Original) The graphics processor of claim 1, wherein the sample portion global state value is a fixed value.

3. (Original) The graphics processor of claim 1, wherein the sample portion global state value is programmed.

4. (Original) The graphics processor of claim 1, wherein the sample portion global state value is determined dynamically.

5. (Previously Presented) A graphics processor for multithreaded execution of program instructions associated with threads to simultaneously process at least two sample types comprising:

a thread control unit including:

an output buffer storing data at one or more output pixel positions as a portion of thread state data;

a thread storage resource (TSR) configured to store the thread state data, the thread state data including a write flag indicating one of the threads will write data to one of the output pixel positions.

6. (Currently Amended) A method of multithreaded processing of at least two sample types of graphics data, each thread in a multithreaded processing unit being configurable to process any of the sample types, comprising:

receiving a plurality of samples of different sample types;

determining a type of the sample;

determining how many of the threads are available for allocation among the sample types;

determining how many of the threads are assignable to each of the sample types; [[and]]

assigning the samples to ~~the thread~~ each of the threads for processing, and simultaneously processing at least some of the plurality of the samples.

7. (Previously Presented) The method of claim 6, further comprising:

determining that a thread is available for assignment to the sample based on a thread execution priority.

8. (Previously Presented) The method of claim 6, further comprising:

determining that a thread is available for assignment to the type of the sample based on allocation of threads among the types of the samples which is fixed or programmable.

9. (Original) The method of claim 6, further comprising:

passing a priority check based on an allocation priority for the sample type prior to assigning the sample to the thread.

10. (Previously Presented) The method of claim 6, further comprising:
receiving another sample;
determining that one of the threads is not available for assignment to the other sample; and
storing the sample while waiting for one of the threads to become available for assignment to the sample.

11. (Previously Presented) A method of multithreaded processing of at least two sample types of graphics data each thread in a multithreaded processing unit being configurable to process any of the sample types comprising:
determining that a thread is available for assignment to a sample;
determining that a sample is available to be processed by a thread;
determining the type of sample that is available; and
assigning the sample that is available to the thread that is available using a thread allocation priority to identify the sample for assignment to the thread that is available.

12. (Previously Presented) The method of claim 11, further comprising:
determining that additional samples are available to be processed by threads; and
simultaneously processing more than one sample of the same type in the threads.

13. (Original) The method of claim 11, further comprising:
determining that a thread is not available for assignment to another sample; and
waiting for a thread to become available for assignment to the sample.

14. (Previously Presented) A graphics processor as claimed in claim 5, wherein one of the sample types is a pixel sample, the thread control unit further being configured to

accept only one pixel sample corresponding to a single position within the output buffer by one of the threads, preventing a position hazard.

15. (Previously Presented) A graphics processor as claimed in claim 14, wherein the thread state data further includes a process independent of order received (PIOR) flag to prevent one of the position hazards in the output buffer.

16. (Previously Presented) The method of claim 12 further comprising associating a first set of instructions with one of the threads and a second set of instructions for another of the threads for operations on the sample assigned to the thread.

17. (Previously Presented) The method of claim 16 further comprising associating separate program counters with each of the threads to allow separate timing of execution of the first and second instruction sets.

18. (Previously Presented) The method of claim 11 wherein the allocation of the threads to processing of the samples is based on a representative size of primitives defined by the graphics data.

19. (Previously Presented) The method of claim 12 further comprising associating a first set of instructions with one of the threads processing one sample type and then associating a second, different set of instructions with the same thread for the next processed sample type.

20. (Previously Presented) The method of claim 19 wherein the samples processed by successive, different instruction sets in the same thread are the same type.